NAME

ieee80211_radiotap - 802.11 device packet capture support

SYNOPSIS

#include <net80211/ieee80211_var.h>

void
ieee80211_radiotap_attach(*struct ieee80211com *, struct ieee80211_radiotap_header *th, int tlen, uint32_t tx_radiotap, struct ieee80211_radiotap_header *rh, int rlen, uint32_t rx_radiotap*);

int
ieee80211_radiotap_active_vap(*struct ieee80211vap **);

int
ieee80211_radiotap_active(*struct ieee80211com **);

void
ieee80211_radiotap_tx(*struct ieee80211vap *, struct mbuf **);

DESCRIPTION

The **net80211** layer used by 802.11 drivers includes support for a device-independent packet capture format called **radiotap** that is understood by tools such as tcpdump(1). This facility is designed for capturing 802.11 traffic, including information that is not part of the normal 802.11 frame structure.

Radiotap was designed to balance the desire for a hardware-independent, extensible capture format against the need to conserve CPU and memory bandwidth on embedded systems. These considerations led to a format consisting of a standard preamble followed by an extensible bitmap indicating the presence of optional capture fields. A **net80211** device driver supporting *radiotap* defines two packed structures that it shares with **net80211**. These structures embed an instance of a *ieee80211_radiotap_header* structure at the beginning, with subsequent fields in the appropriate order, and macros to set the bits of the *it_present* bitmap to indicate which fields exist and are filled in by the driver. This information is then supplied through the **ieee80211_radiotap_attach**() call after a successful **ieee80211_ifattach**() request.

With radiotap setup, drivers just need to fill in per-packet capture state for frames sent/received and dispatch capture state in the transmit path (since control is not returned to the **net80211** layer before the packet is handed to the device). To minimize overhead this work should be done only when one or more processes are actively capturing data; this is checked with one of **ieee80211_radiotap_active_vap**() and **ieee80211_radiotap_active**(). In the transmit path capture work looks like this:

```
if (ieee80211_radiotap_active_vap(vap)) {
        ... /* record transmit state */
        ieee80211_radiotap_tx(vap, m); /* capture transmit event */
}
```

While in the receive path capture is handled in **net80211** but state must be captured before dispatching a frame:

```
if (ieee80211_radiotap_active(ic)) {
        ... /* record receive state */
}
...
ieee80211_input(...);           /* packet capture handled in net80211 */
```

The following fields are defined for *radiotap*, in the order in which they should appear in the buffer supplied to **net80211**.

IEEE80211_RADIOTAP_TSFT
: This field contains the unsigned 64-bit value, in microseconds, of the MAC's 802.11 Time Synchronization Function (TSF). In theory, for each received frame, this value is recorded when the first bit of the MPDU arrived at the MAC. In practice, hardware snapshots the TSF otherwise and one cannot assume this data is accurate without driver adjustment.

IEEE80211_RADIOTAP_FLAGS
: This field contains a single unsigned 8-bit value, containing one or more of these bit flags:

  IEEE80211_RADIOTAP_F_CFP
  : Frame was sent/received during the Contention Free Period (CFP).

  IEEE80211_RADIOTAP_F_SHORTPRE
  : Frame was sent/received with short preamble.

  IEEE80211_RADIOTAP_F_WEP
  : Frame was encrypted.

  IEEE80211_RADIOTAP_F_FRAG
  : Frame was an 802.11 fragment.

  IEEE80211_RADIOTAP_F_FCS
  : Frame contents includes the FCS.

IEEE80211_RADIOTAP_F_DATAPAD
: Frame contents potentially has padding between the 802.11 header and the data payload to align the payload to a 32-bit boundary.

IEEE80211_RADIOTAP_F_BADFCS
: Frame was received with an invalid FCS.

IEEE80211_RADIOTAP_F_SHORTGI
: Frame was sent/received with Short Guard Interval.

IEEE80211_RADIOTAP_RATE
: This field contains a single unsigned 8-bit value that is the data rate. Legacy rates are in units of 500Kbps. MCS rates (used on 802.11n/HT channels) have the high bit set and the MCS in the low 7 bits.

IEEE80211_RADIOTAP_CHANNEL
: This field contains two unsigned 16-bit values. The first value is the center frequency for the channel the frame was sent/received on. The second value is a bitmap containing flags that specify channel properties.

  This field is deprecated in favor of IEEE80211_RADIOTAP_XCHANNEL but may be used to save space in the capture file for legacy devices.

IEEE80211_RADIOTAP_DBM_ANTSIGNAL
: This field contains a single signed 8-bit value that indicates the RF signal power at the antenna, in decibels difference from 1mW.

IEEE80211_RADIOTAP_DBM_ANTNOISE
: This field contains a single signed 8-bit value that indicates the RF noise power at the antenna, in decibels difference from 1mW.

IEEE80211_RADIOTAP_DBM_TX_POWER
: Transmit power expressed as decibels from a 1mW reference. This field is a single signed 8-bit value. This is the absolute power level measured at the antenna port.

IEEE80211_RADIOTAP_ANTENNA
: This field contains a single unsigned 8-bit value that specifies which antenna was used to transmit or receive the frame. Antenna numbering is device-specific but typically the primary antenna has the lowest number. On transmit a value of zero may be seen which typically means antenna selection is left to the device.

IEEE80211_RADIOTAP_DB_ANTSIGNAL
: This field contains a single unsigned 8-bit value that indicates the RF signal power at the antenna, in decibels difference from an arbitrary, fixed reference.

IEEE80211_RADIOTAP_DB_ANTNOISE
: This field contains a single unsigned 8-bit value that indicates the RF noise power at the antenna, in decibels difference from an arbitrary, fixed reference.

IEEE80211_RADIOTAP_XCHANNEL
: This field contains four values: a 32-bit unsigned bitmap of flags that describe the channel attributes, a 16-bit unsigned frequency in MHz (typically the channel center), an 8-bit unsigned IEEE channel number, and a signed 8-bit value that holds the maximum regulatory transmit power cap in .5 dBm (8 bytes total). Channel flags are defined in: *<net80211/_ieee80211.h>* (only a subset are found in *<net80211/ieee80211_radiotap.h>*). This property supersedes IEEE80211_RADIOTAP_CHANNEL and is the only way to completely express all channel attributes and the mapping between channel frequency and IEEE channel number.

EXAMPLES

Radiotap receive definitions for the Intersil Prism driver:

```
#define WI_RX_RADIOTAP_PRESENT \
     ((1 << IEEE80211_RADIOTAP_TSFT) \
      (1 << IEEE80211_RADIOTAP_FLAGS) | \
      (1 << IEEE80211_RADIOTAP_RATE) | \
      (1 << IEEE80211_RADIOTAP_CHANNEL) | \
      (1 << IEEE80211_RADIOTAP_DB_ANTSIGNAL) | \
      (1 << IEEE80211_RADIOTAP_DB_ANTNOISE))

struct wi_rx_radiotap_header {
     struct ieee80211_radiotap_header wr_ihdr;
     uint64_t       wr_tsf;
     uint8_t        wr_flags;
     uint8_t        wr_rate;
     uint16_t       wr_chan_freq;
     uint16_t       wr_chan_flags;
     uint8_t        wr_antsignal;
     uint8_t        wr_antnoise;
} __packed __aligned(8);
```

and transmit definitions for the Atheros driver:

```
#define ATH_TX_RADIOTAP_PRESENT (                \
        (1 << IEEE80211_RADIOTAP_FLAGS)       | \
        (1 << IEEE80211_RADIOTAP_RATE)        | \
        (1 << IEEE80211_RADIOTAP_DBM_TX_POWER) | \
        (1 << IEEE80211_RADIOTAP_ANTENNA)     | \
        (1 << IEEE80211_RADIOTAP_XCHANNEL)    | \
        0)

struct ath_tx_radiotap_header {
        struct ieee80211_radiotap_header wt_ihdr;
        uint8_t       wt_flags;
        uint8_t       wt_rate;
        uint8_t       wt_txpower;
        uint8_t       wt_antenna;
        uint32_t      wt_chan_flags;
        uint16_t      wt_chan_freq;
        uint8_t       wt_chan_ieee;
        int8_t        wt_chan_maxpow;
} __packed;
```

SEE ALSO

tcpdump(1), bpf(4), ieee80211(9)

HISTORY

The **ieee80211_radiotap** definitions first appeared in NetBSD 1.5.

AUTHORS

The original version of this manual page was written by Bruce M. Simpson <*bms@FreeBSD.org*> and Darron Broad <*darron@kewl.org*>.